



# Programming Fundamental

## Week 11



---

### Learning Outcomes:

After this lesson, students will be able to:

1. Know how to declare and use multidimensional compile-time arrays.
2. Know how to use implied do loops in I/O statements for input and output of multidimensional arrays.
3. Know how to specify array sections and sub arrays of multidimensional arrays.
4. Be able to use the *transpose* array function.
5. Be able to use and interpret multidimensional array expressions.

### Instructions

- Use proper indentation to make your programs readable.
- Use descriptive variables in your programs (Name of the variables should show their purposes)

### 2D (2 Dimensional array):

The two-dimensional array can be defined as an array of arrays. The 2D array is organized as matrices which can be represented as the collection of rows and columns.

#### Visual Representation of 2dArray:

Two dimensional array with 3 rows and 4 columns indexes.

	Column 1	Column 2	Column 3	Column 4
Row 1	<code>x[0][0]</code>	<code>x[0][1]</code>	<code>x[0][2]</code>	<code>x[0][3]</code>
Row 2	<code>x[1][0]</code>	<code>x[1][1]</code>	<code>x[1][2]</code>	<code>x[1][3]</code>
Row 3	<code>x[2][0]</code>	<code>x[2][1]</code>	<code>x[2][2]</code>	<code>x[2][3]</code>

#### Declaration of Two Dimensional Array in C:

The syntax to declare the 2D array is given below.

```
data_type array_name[rows][columns];
```

Consider the following example.

```
int twodimen[4][3];
```

Here, 4 is the number of rows, and 3 is the number of columns.

## Initialization of 2D Array in C++

In the 1D array, we don't need to specify the size of the array if the declaration and initialization are being done simultaneously. However, this will not work with 2D arrays. We will have to define at least the second dimension of the array. The two-dimensional array can be declared and defined in the following way.

```
int arr[4][3]={{1,2,3},{2,3,4},{3,4,5},{4,5,6}};
```

```
int num[3][3]={{ { 25,10,5},{4,6,13},{45,90,78} };
```

```
int num[3][3]={{ { 25,10,5}, //row0
```

```
{ 4,6,13}, //row1
```

```
{ 45,90,78} //row2
```

```
};
```

```
//this format also stores data in the array but results in a loss of readability
```

```
int num[2][4]={4,6,8,10,3,5,7,9};
```

## Figure-1:-

	Red	Black	Brown	Blue	Gray
Suzuki	10	7	12	10	4
Toyota	18	11	15	17	2
Nissan	23	19	12	16	14
BMW	7	12	16	0	2
Audi	3	5	6	2	1



**Note:** We will perform different operation on Figure-1, So keep Figure-1 in mind and focus on problem statement.

**Example #1:**

**Write a program that initialize matrix named as cars mentioned in Figure-1 and display values.**

<b>Solution</b>
-----------------

```

#include<iostream>
using namespace std;

main()
{
    int cars[5][5] = {{10, 7, 12, 10, 4},
                      {18, 11, 15, 17, 2},
                      {23, 19, 12, 16, 14},
                      {7, 12, 16, 0, 2},
                      {3, 5, 6, 2, 1}};

    for(int x = 0; x < 5; x = x + 1)
    {
        for(int y = 0; y < 5; y = y + 1)
        {
            cout << cars[x][y] << "\t";
        }
        cout << endl;
    }
}

```

The code produces the following output

10	7	12	10	4
18	11	15	17	2
23	19	12	16	14
7	12	16	0	2
3	5	6	2	1

**Example #2:**

Write a program that initialize of cars as shown in Figure-1 and display the number of Toyota Blue cars available in carriage.

### Solution

```
#include<iostream>
using namespace std;

main()
{
    int cars[5][5] = {{10, 7, 12, 10, 4},
                      {18, 11, 15, 17, 2},
                      {23, 19, 12, 16, 14},
                      {7, 12, 16, 0, 2},
                      {3, 5, 6, 2, 1}
    };

    cout <<"Toyota Blue Cars=" <<cars[1][3];
}
```

The code produces the following output

```
Toyota Blue Cars=17
```

### Example #3:

Write a Program that display total number of “Red” cars in carriage as shown in Figure-1 without using loop.


#### Solution

```
#include<iostream>
using namespace std;

main()
{
    int sum;
    int cars[5][5] = {{10, 7, 12, 10, 4},
                     {18, 11, 15, 17, 2},
                     {23, 19, 12, 16, 14},
                     {7, 12, 16, 0, 2},
                     {3, 5, 6, 2, 1}
                    };

    sum=cars[0][0]+cars[1][0]+cars[2][0]+cars[3][0]+cars[4][0];
    cout <<"Total Number Red cars=" <<sum<<endl;
}
```

The code produces the following output

 E:\PF new\cars\_red\_sum.exe

Total Number Red cars=61

#### Example#4

Write a Program that display total number of “Nissan” cars in carriage as shown in Figure-1 using loop.

Solution
<pre>#include&lt;iostream&gt; using namespace std;  main() {     int sum=0;     int cars[5][5] = {{10, 7, 12, 10, 4},                       {18, 11, 15, 17, 2},                       {23, 19, 12, 16, 14},                       {7, 12, 16, 0, 2},                       {3, 5, 6, 2, 1}                       };     for(int i=0;i&lt;5;i++)         sum=sum+cars[0][i];      cout &lt;&lt;"Total Number Nissan cars=" &lt;&lt;sum&lt;&lt;endl; }</pre>
The code produces the following output
<pre>Total Number Nissan cars=43</pre>

#### Example#5

Write a program that pass color as parameter into function named as “converter” and return index of that column and sum total number of cars of that color availbe.

## Solution

```
#include<iostream>
using namespace std;
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                  };
string colr;
int converter(string color)
{
    int index;
    if(color=="red")
        index=0;
    else if(color=="black")
        index=1;
    else if(color=="brown")
        index=2;
    else if(color=="blue")
        index=3;
    else if(color=="gray")
        index=4;
    else
        index=-1;

    return index;
}
```



```

int total_cars()
{
    int sum=0,index;
    cout<<"enter color for finding total number of cars"<<endl;
    cin>>colr;
    index=converter(colr);

    for(int i=0;i<5;i++)
        sum=sum+cars[index][i];

    return sum;
}
void show()
{
    cout<<"Total Cars of"<<colr<<"="<<total_cars()<<endl;
}
main()
{
    show();
}

```

**The code produces the following output**

```

enter color for finding total number of cars
black
Total Cars ofblack=63

```

### Example#6

Write a program that calculates total number of cars of Figure-1 using function that return sum of total cars.

#### Solution

```
#include<iostream>
using namespace std;
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
};

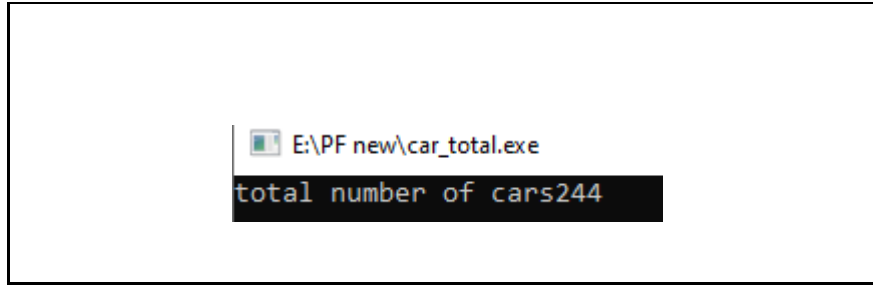
int total_sum()
{
    int sum=0;

    for(int i=0;i<5;i++)
        for(int j=0;j<5;j++)
            sum=sum+cars[i][j];
    return sum;
}

void show()
{
    cout<<"total number of cars"<<total_sum();
}

main()
{
    show();
}
```

The code produces the following output



### Example#7

Write program that change the position of color with position of car's company of

Figure-1 , output should be as given below

		Suzuki	Toyota	Nissan	BMW	Audi
		0	1	2	3	4
Red	0	10	18	23	7	3
Black	1	7	11	19	12	5
Brown	2	12	15	12	16	6
Blue	3	10	17	16	0	2
Gray	4	4	2	14	2	1

**Solution**

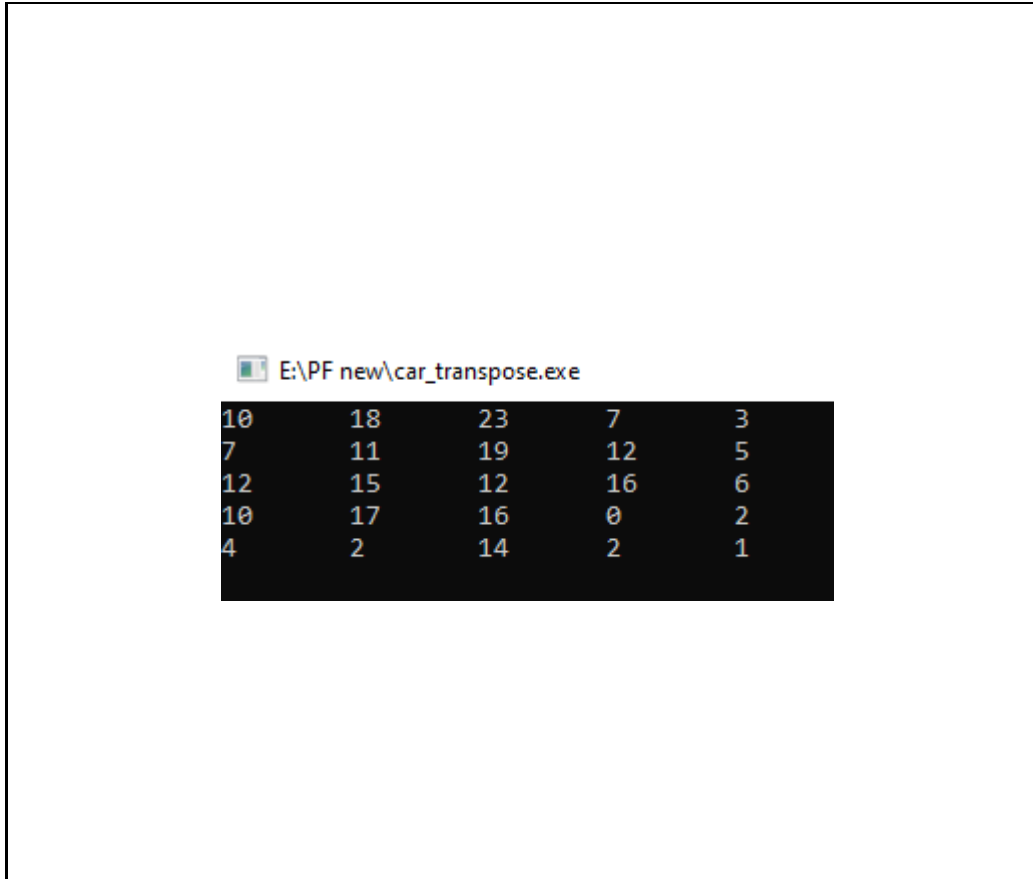
```

#include<iostream>
using namespace std;
int cars[5][5] = {{10, 7, 12, 10, 4},
                  {18, 11, 15, 17, 2},
                  {23, 19, 12, 16, 14},
                  {7, 12, 16, 0, 2},
                  {3, 5, 6, 2, 1}
                  };

main()
{
    for(int i=0;i<5;i++)
    {
        for(int j=0;j<5;j++)
        {
            cout<<cars[j][i]<<"\t";
            cout<<"\n";
        }
    }
}

```

**The code produces the following output**



### Challenge#1:

Write a program that read 3x3 matrix and show sum of element of that matrix.

#### Input:

2	4	5
6	7	1
0	1	2

Output:

**28**

### Challenge#2:

Write a program that read 3x3 matrix and check whether the matrix is identity matrix or not.

**Note:** the identity matrix of size n is the  $n \times n$  square matrix with ones on the main diagonal and zeros elsewhere

### Challenge#3:

Remember the game Battleship? Ships are floating in a matrix. You have to fire torpedoes at their suspected coordinates, to try and hit them.

Create a function that takes a coordinate as a string. If the coordinate contains only water ".", return "splash" and if the coordinate contains a ship "\*", return "BOOM".

#### Notes:

- The provided matrix is always a square.
- The provided matrix will not be larger than 5 \* 5 ( A1 \* E5).

#### Test Cases:

```
[
  [ " ", " ", " ", " ", " * ", " * " ],
  [ " ", " ", " ", " ", " ", " " ],
  [ " ", " * ", " ", " ", " ", " " ],
  [ " ", " * ", " ", " ", " ", " " ],
  [ " ", " * ", " ", " ", " ", " " ],
  [ " ", " * ", " ", " ", " ", " " ],
  [ " ", " ", " ", " * ", " * ", " " ],
]
```

fire("A1") → "splash"

fire("A4") → "BOOM"

fire("D2") → "BOOM"

#### Challenge#4 :

In (American) Football, a team can score if they manage to kick a ball through the goal (i.e. above the crossbar and between the uprights).

Create a function that returns true if the ball 0 goes through the goal. You will be given a 2D array.

Note:

- All goals will be of the same size.
- All arrays will be 7 x 16.
- A team can never score if it hits the crossbar or goes underneath it.

Test Cases:

•

```
[
  [ " #      # " ],
  [ " # 0  # " ],
  [ " #      # " ],
  [ " ##### " ],
]
```

```
[" # "],
[" # "],
[" # "]
]
isGoalScore() → true
```

•

```
[
[" # 0 # "]
[" # # "],
[" # # "],
[" ##### "],
[" # "],
[" # "],
[" # "]
]
isGoalScore() → true
```

•

```
[
[" # # "]
[" # # "],
[" # # "],
[" ##### "],
[" # 0"],
[" # "],
[" # "]
]
isGoalScore() → false
```

### Challenge#5:

Create a function that checks 4x3 2D array and returns a count of the total number of identical rows.

### Test Cases:

- `[[0, 0, 0], [0, 1, 2], [0, 0, 0], [2, 1, 0]]`  
`countIdenticalArrays() → 2`
- `[0, 1, 0], [0, 1, 2], [0, 2, 0], [2, 1, 0]`

`countIdenticalArrays()` → 0

- `[0, 1, 2], [0, 1, 2], [0, 1, 2], [2, 1, 0]`  
`countIdenticalArrays()` → 3